RecentInserts

Quick Start

If you have a number of users that periodically need to download files recently added/inserted into your data holdings, this script may be useful. It can be used to periodically go through a configured file system to find the recently inserted files, and generate listings of them. By making these listings available, the users do not need to iterate through your file system remotely.

You can download the script by cloning the repository using the following command:

```
> git clone https://git.earthdata.nasa.gov/scm/hdd/recentinserts.git recent-inserts
```

or you may browse the repository on-line at:

https://git.earthdata.nasa.gov/projects/HDD/repos/recentinserts/browse

Introduction

One of the problems with providing on-line access to data holdings is that users are often interested only in downloading recently inserted files. Whether data access is provided by FTP or HTTP, this usually requires scanning the entire directory tree (or the relevant sections of it) one directory at a time to find these files. This document outlines an approach for automatically providing lists of files recently inserted into publicly accessible data holdings. By providing, periodic, easily identifiable file lists, we can significantly reduce the data server loads that would otherwise be needed to scan and compile these lists on a user-by-user basis.

Method

The approach recommended here is to use a cron task that is periodically invoked. This task will scan the appropriate directories looking for files inserted between specific time intervals and build up a list of the file names. The resulting file list will be prominently named and located so that clients can easily identify and retrieve them (either manually, or using an automation script).

A key point to note is that the task must scan for files using specific start and end times. This is because scanning large file systems can take a minute or two, and files can be added to a directory after it has been scanned. Without specific start and end times, files will either be missed or duplicated between scans.

Additionally, the user under which the task executes must have the necessary directory read and file create permissions, otherwise an error will occur.

The sample script below shows a bash shell implementation of a recent-inserts file generator. Three key configuration values are needed:

1. The directory to scan for new files
2. The interval at which to scan the directory
3. The location and name of the file listing to generate

The script works by first finding the last complete time interval that has elapsed (thus the configured interval must divide evenly into a 24 hour period). It then creates two temporary files with timestamps bounding the interval and initiates a scans of the configured directory using those two files to constrain the search. The resulting files are sorted and copied into the configured output file. For reference purposes, it will also output the start and end time of the scan into the file listing.

Setup

When configuring the script to run as a cron task, it is important to coordinate the execution time with the time interval configured in the script. The script should be invoked shortly after each time interval starting from midnight. For example, if the configured time interval is 30 minutes, the cron task should execute a minute or two after each hour and half hour. A typical crontab entry for this might be:

```
1,31 * * * * /tools/script/recent_inserts.sh
```

For a time interval of 15 minutes, a typical crontab entry would be:

```
1,16,31,46 * * * * /tools/script/recent_inserts.sh
```

In order to prevent a build-up of the generated file lists, it will be necessary to clean old lists. This can be done very simply with another cron task that executes periodically (e.g. once daily). The following example checks at 5 minutes past midnight for files generated by the above script older than 7 days, and removes them.

```
5 0 * * * /usr/bin/find "/mnt/data_holdings/FS1/RecentInserts/RecentInserts_*.txt" -type f -mtime
+7 -exec rm -f {} \;
```